Métodos Exatos

Pedro Hokama

Fontes

- Curso Discrete Optimization no Coursera - Prof. Dr. Pascal Van Hentenryck
- [*clrs*] Algoritmos: Teoria e Prática (Terceira Edição) Thomas H. Cormen, Charles Eric Leiserson, Ronald Rivest e Clifford Stein.
- [*timr*] Algorithms Illuminated Series, Tim Roughgarden
- Pesquisa Operacional, Arenales, M. N., Armentano, V. A., Morabito Neto, R., e Yanasse, H. H. (2015)
- Algoritmos, Sanjoy Dasgupta, Christos Papadimitriou e Umesh Vazirani
- Stanford Algorithms
  `https://www.youtube.com/playlist?list=PLXFMmlk03Dt7QOxr1PIAriY5623cKiH7V`
  `https://www.youtube.com/playlist?list=PLXFMmlk03Dt5EMI2s2WQBsLsZl7A5HEK6`
- Introdução à Otimização Combinatória, Flávio K. Miyazawa e Cid C. de Souza.
- Otimização Combinatória, da Carla Negri Lintzmayer do CMCC, UFABC

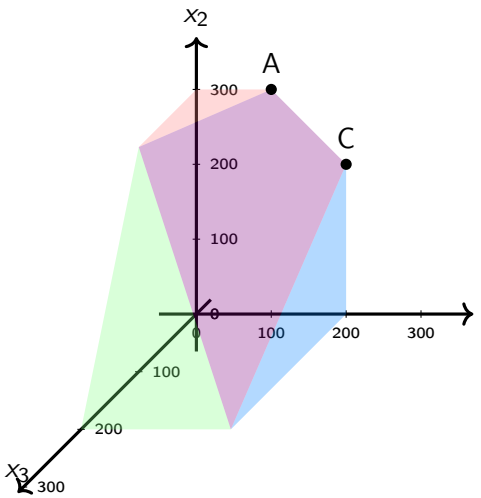
Qualquer erro é de minha responsabilidade.

- George Dantzig (1914 - 2005) foi um cientista matemático norte-americano.
- Fez várias contribuições na engenharia, pesquisa operacional, computação e matemática. Ao chegar atrasado em uma aula, resolveu dois problemas estatísticos, até então em aberto, confundido-os com lição de casa.
- Inventou o algoritmo simplex para Programação Linear em 1947

Algoritmo Simplex

- Vamos começar agora a ver o algoritmo simplex. Primeiramente daremos algumas intuições e definições.
- No espaço $\mathbb{R}^n$ cada equação linear define um **hiperplano**,
- e cada inequação define um **semi-espaço**,
- a região factível é dada pela intersecção dos semi-espaços,
- e forma um **poliedro convexo** (politopo se for limitado).

- Como o poliedro é convexo, um ótimo local também é um ótimo global.



max
$x_1 + 6x_2 + 13x_3$
sujeito a

$$\begin{aligned} x_1 &\leq 200 && (1)\\ x_2 &\leq 300 && (2)\\ x_1 + x_2 + x_3 &\leq 400 && (3)\\ x_2 + 3x_3 &\leq 600 && (4)\\ x_1 &\geq 0 && (5)\\ x_2 &\geq 0 && (6)\\ x_3 &\geq 0 && (7) \end{aligned}$$

- O vértice é um ponto único onde um subconjunto de hiperplanos se encontra.
- O ponto A é o único ponto no qual (2), (3) e (7) são satisfeitas com igualdade.
- Por outro lado os hiperplanos (4) e (6) não definem um vértice, porque sua interseção é uma linha inteira.

Selecione um subconjunto das inequações. Se existir um único ponto que satisfaça todas elas com igualdade, e este ponto for factível, então ele será um vértice.

- Se temos n variáveis, precisamos de pelo menos n equações lineares se queremos uma única solução.
- Mais do que n equações é redundante, e pelo menos uma delas pode ser reescrita como combinação linear das outras, e pode ser descartada.

Cada vértice é especificado por um conjunto de n inequações.

- Uma noção de vizinho segue naturalmente

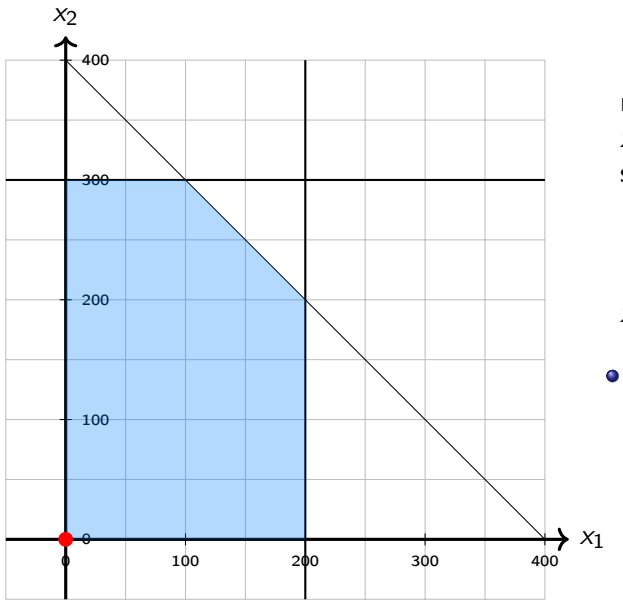
Dois vértices são vizinhos se eles têm em comum $n-1$ das inequações que os definem.

- Por exemplo, na nossa figura, A e C compartilham as inequações (3) e (7).

- O algoritmo simplex trabalha com um sistema de equações, dessa forma devemos transformar o nosso PL.

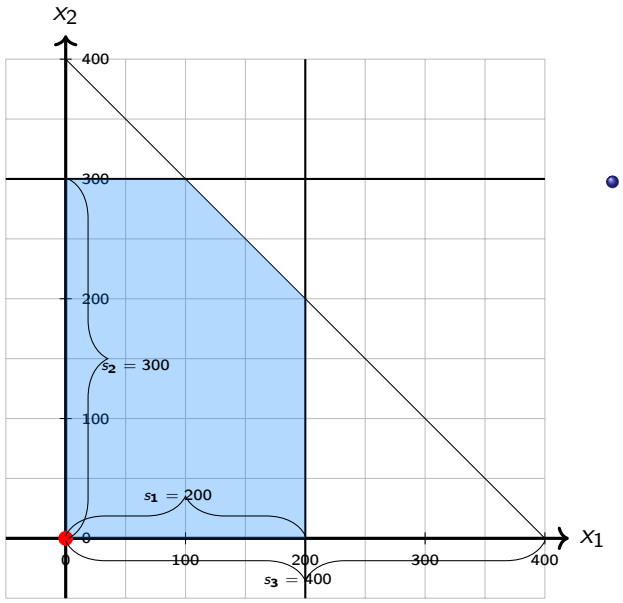
$$\begin{aligned} \max \quad & x_1 + 6x_2 & & \\ \text{sujeito a} \quad & x_1 + s_1 & \leq & 200 \\ & x_2 + s_2 & \leq & 300 \\ & x_1 + x_2 + s_3 & \leq & 400 \\ & x_1, x_2, s_1, s_2, s_3 & \geq & 0 \end{aligned}$$

- Iremos dividir nossas variáveis em duas partes, vamos chamar de variáveis **na base** aquelas com algum valor e variáveis **fora da base** aquelas que são iguais a zero.
- Escrevemos as variáveis na base em função das variáveis fora da base.
- Vamos começar com as variáveis de folga na base e as variáveis x_1 e x_2 fora da base (ou seja, iguais a zero, e portanto estamos falando do vértice na origem).



maximizar
$x_1 + 6x_2$
sujeito a
$s_1 = 200 - x_1$
$s_2 = 300 - x_2$
$s_3 = 400 - x_1 - x_2$
$x_1, x_2, s_1, s_2, s_3 \geq 0$

- agora olhe para a função objetivo, alguma tem coeficiente positivo? Isso significa que vale a pena aumentar aquela variável.



- intuitivamente o valor das variáveis de folga (com os devidos ajustes) significa a distância que estamos de cada uma das restrições

- como x_1 tem coeficiente positivo na função objetivo, vamos aumentar o valor dela (colocar ela na base), mas para isso precisamos tirar alguém da base.
- Precisamos descobrir qual é a variável que pode sair da base.

O que aconteceria se escolhessemos s_3 para sair da base?
maximizar
$x_1 + 6x_2$
sujeito a

$$\begin{aligned} s_1 &= 200 - x_1 \\ s_2 &= 300 - x_2 \\ s_3 &= 400 - x_1 - x_2 \end{aligned}$$

$x_1, x_2, s_1, s_2, s_3 \geq 0$

O que aconteceria se escolhessemos s_3 para sair da base?
maximizar
$(400 - s_3 - x_2) + 6x_2$
sujeito a

$$\begin{aligned} s_1 &= 200 - (400 - s_3 - x_2) \\ s_2 &= 300 - x_2 \\ x_1 &= 400 - s_3 - x_2 \end{aligned}$$

$x_1, x_2, s_1, s_2, s_3 \geq 0$

O que aconteceria se escolhessemos s_3 para sair da base?
maximizar
$(400 - s_3 - x_2) + 6x_2$
sujeito a

$$\begin{aligned} s_1 &= -200 + s_3 + x_2 \\ s_2 &= 300 - x_2 \\ x_1 &= 400 - s_3 - x_2 \end{aligned}$$

$x_1, x_2, s_1, s_2, s_3 \geq 0$

maximizar
$(400 - s_3 - x_2) + 6x_2$
sujeito a

$$\begin{aligned} s_1 &= -200 + s_3 + x_2 \\ s_2 &= 300 - x_2 \\ x_1 &= 400 - s_3 - x_2 \end{aligned}$$

$x_1, x_2, s_1, s_2, s_3 \geq 0$
Obteriamos uma solução não basica (inviável) pois estamos saindo da região viável.
A variável s_2 também não pode ser escolhida pois x_1 não aparece na segunda equação. A única escolha então é s_1

- Podemos simplificar a ideia, calculando os seguintes valores:
- seja b_i o valor da constante da equação i, e seja a_{i1} o valor do coeficiente de x_1 na equação i. Vamos procurar o $\frac{b_i}{-a_{i1}}$ mínimo.

maximizar
$x_1 + 6x_2$
sujeito a

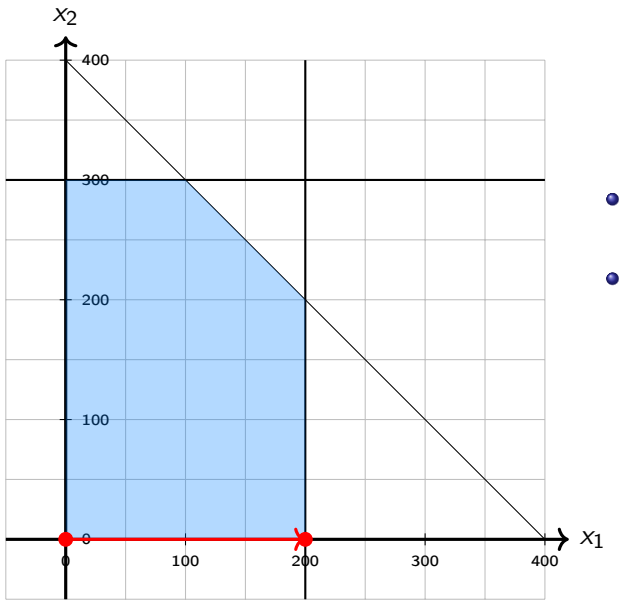
$$\begin{aligned} s_1 &= 200 - x_1 && \frac{b_1}{-a_{11}} = \frac{200}{-(-1)} = 200 \\ s_2 &= 300 - x_2 && \frac{b_2}{-a_{21}} = \frac{300}{0} \\ s_3 &= 400 - x_1 - x_2 && \frac{b_3}{-a_{31}} = \frac{400}{-(-1)} = 400 \end{aligned}$$

$x_1, x_2, s_1, s_2, s_3 \geq 0$

maximizar
$(200 - s_1) + 6x_2$
sujeito a

$$\begin{aligned} x_1 &= 200 - s_1 \\ s_2 &= 300 - x_2 \\ s_3 &= 400 - (200 - s_1) - x_2 \end{aligned}$$

$x_1, x_2, s_1, s_2, s_3 \geq 0$



- x_1 era zero, foi para 200
- note que se escolhêssemos s_3 para sair da base, sairíamos da região viável.

maximizar
$200 - s_1 + 6x_2$
sujeito a

$$\begin{aligned} x_1 &= 200 - s_1 & \frac{b_1}{-a_{12}} &= \frac{200}{0} \\ s_2 &= 300 - x_2 & \frac{b_2}{-a_{22}} &= \frac{300}{-(-1)} = 300 \\ s_3 &= 200 + s_1 - x_2 & \frac{b_3}{-a_{32}} &= \frac{200}{-(-1)} = 200 \end{aligned}$$

$x_1, x_2, s_1, s_2, s_3 \geq 0$

maximizar
$200 - s_1 + 6x_2$
sujeito a

$$\begin{aligned} x_1 &= 200 - s_1 \\ s_2 &= 300 - x_2 \\ x_2 &= 200 + s_1 - s_3 \end{aligned}$$

$x_1, x_2, s_1, s_2, s_3 \geq 0$

maximizar
$200 - s_1 + 6(200 + s_1 - s_3)$
sujeito a

$$\begin{aligned} x_1 &= 200 - s_1 \\ s_2 &= 300 - (200 + s_1 - s_3) \\ x_2 &= 200 + s_1 - s_3 \end{aligned}$$

$x_1, x_2, s_1, s_2, s_3 \geq 0$

maximizar
$200 - s_1 + 1200 + 6s_1 - 6s_3$
sujeito a

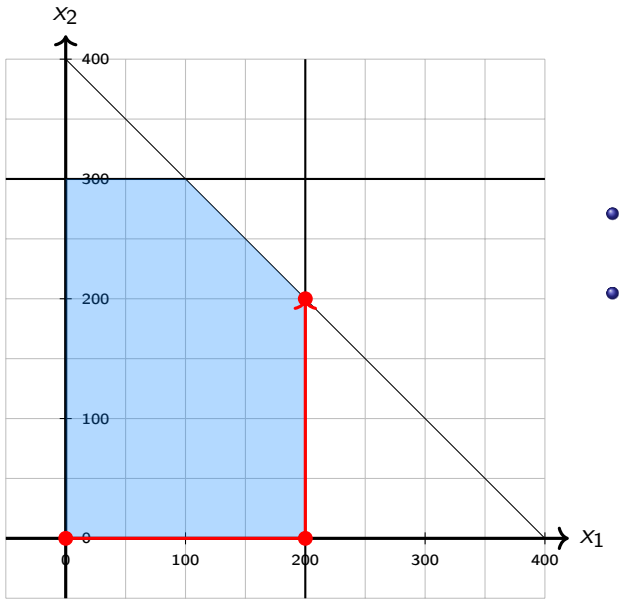
$$\begin{aligned} x_1 &= 200 - s_1 \\ s_2 &= 300 - 200 - s_1 + s_3 \\ x_2 &= 200 + s_1 - s_3 \end{aligned}$$

$x_1, x_2, s_1, s_2, s_3 \geq 0$

maximizar
$1400 + 5s_1 - 6s_3$
sujeito a

$$\begin{aligned} x_1 &= 200 - s_1 \\ s_2 &= 100 - s_1 + s_3 \\ x_2 &= 200 + s_1 - s_3 \end{aligned}$$

$x_1, x_2, s_1, s_2, s_3 \geq 0$



- x_2 era zero, foi para 200
- note que tirar s_3 da base, significa zerar a distância até a restrição 3.

maximizar
$1400 + 5s_1 - 6s_3$ coef. de s_1 é positivo
sujeito a

$$\begin{aligned} x_1 &= 200 - s_1 & \frac{200}{-(-1)} &= 200 \\ s_2 &= 100 - s_1 + s_3 & \frac{100}{-(-1)} &= 100 \\ x_2 &= 200 + s_1 - s_3 & \frac{200}{-1} &= -200 \end{aligned}$$

$x_1, x_2, s_1, s_2, s_3 \geq 0$
precisamos escolher o menor coeficiente positivo

maximizar
$1400 + 5s_1 - 6s_3$
sujeito a

$$\begin{aligned} x_1 &= 200 - s_1 \\ s_1 &= 100 - s_2 + s_3 \\ x_2 &= 200 + s_1 - s_3 \end{aligned}$$

$x_1, x_2, s_1, s_2, s_3 \geq 0$

maximizar
$1400 + 5(100 - s_2 + s_3) - 6s_3$
sujeito a

$$\begin{aligned} x_1 &= 200 - (100 - s_2 + s_3) \\ s_1 &= 100 - s_2 + s_3 \\ x_2 &= 200 + (100 - s_2 + s_3) - s_3 \end{aligned}$$

$x_1, x_2, s_1, s_2, s_3 \geq 0$

maximizar
$1400 + 500 - 5s_2 + 5s_3 - 6s_3$
sujeito a

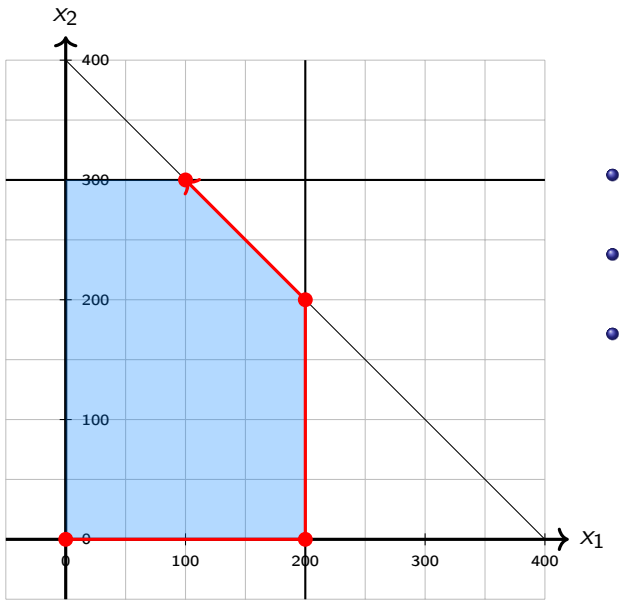
$$\begin{aligned} x_1 &= 200 - 100 + s_2 - s_3 \\ s_1 &= 100 - s_2 + s_3 \\ x_2 &= 200 + 100 - s_2 + s_3 - s_3 \end{aligned}$$

$x_1, x_2, s_1, s_2, s_3 \geq 0$

maximizar
$1900 - 5s_2 - s_3$
sujeito a

$$\begin{aligned} x_1 &= 100 + s_2 - s_3 \\ s_1 &= 100 - s_2 + s_3 \\ x_2 &= 300 - s_2 \end{aligned}$$

$x_1, x_2, s_1, s_2, s_3 \geq 0$



- x_1 era 200, foi para 100
- x_2 era 200, foi para 300
- note que tirar s_2 da base, significa zerar a distância até a restrição 2.

como nenhum coef. é positivo não vale a pena aumentar
maximizar
$1900 - 5s_2 - s_3$
sujeito a

$$\begin{aligned} x_1 &= 100 + s_2 - s_3 \\ s_1 &= 100 - s_2 + s_3 \\ x_2 &= 300 - s_2 \end{aligned}$$

$x_1, x_2, s_1, s_2, s_3 \geq 0$

O algoritmo simplex

Algoritmo 1: Simplex

1 **enquanto** $\exists$ *variável* $y | c_y > 0$ **faça**
2 escolha uma variável y com $c_y > 0$;
3 $l = \arg\min_{i | a_{ie} < 0} \frac{b_i}{-a_{ie}}$;
4 $pivot(y, l)$;

Condições

maximizar
$x_1 + 6x_2$
sujeito a
$s_1 = 200 + x_1$

$s_2 = 300 - x_2$

$s_3 = 400 + x_1 - x_2$

$x_1, x_2, s_1, s_2, s_3 \geq 0$
Note que nesse modelo podemos aumentar x_1 arbitrariamente.Então o programa é dito ilimitado. Em uma aplicação real, isso possivelmente quer dizer que seu modelo está errado.

maximizar
$x_1 + 6x_2$
sujeito a
$s_1 = 0 - x_1$

$s_2 = 300 - x_2$

$s_3 = 400 - x_1 - x_2$

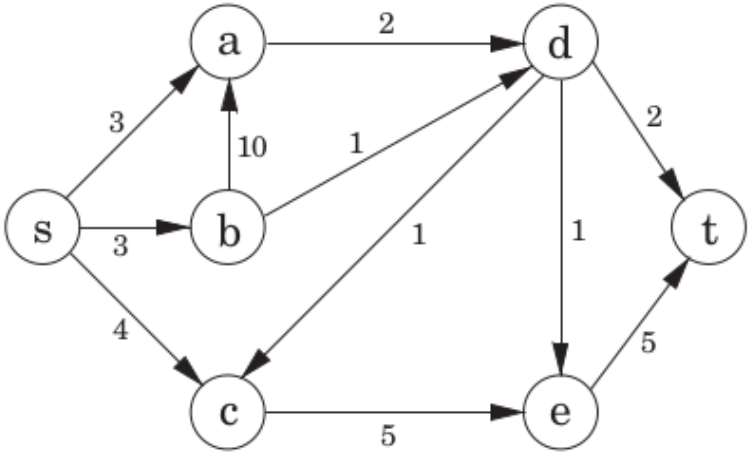
$x_1, x_2, s_1, s_2, s_3 \geq 0$
Se uma variável básica tiver valor 0, tirar ela da base, não vai mudar o valor da função objetivo. Dessa forma o algoritmo ficaria ciclando entre esses valores. Precisamos criar uma nova regra para garantir que o algoritmo termine.

- Regra de Bland.
- Regra de pivoteamento Lexicografico.
- Métodos de perturbação.

Exercício: Modelar o seguinte problema:

Problema do fluxo máximo

Dado um grado direcionado $D = (V, A)$ em que cada arco $(u, v) \in A$ tem uma capacidade $c_{(u,v)}$, um vértice fonte $s \in V$ e um vértice sorvedouro $t \in V$.
Desejamos encontrar um fluxo máximo de s até t, sendo que em todos os vértices exceto s e t a quantidade de fluxo que entra deve ser igual a quantidade de fluxo que sai, e as capacidades dos arcos são respeitadas.



Bibliotecas

- Felizmente existem várias bibliotecas que implementam o simplex (e várias formas de resolução) para resolver PLs (e outras modelagens matemáticas)
- Talvez as mais conhecidas sejam:
  - IBM CPLEX Optimization Studio (comercial)
  - Gurobi Solver (comercial)
  - Google OR-Tools (open source)
  - GLPK (open source)

OR-Tools

- É uma ferramenta de código aberto para otimização combinatória.
- Em geral, o objetivo é encontrar a melhor solução para um problema em um conjunto muito grande de soluções possíveis.

- Exemplos de problemas que podem ser resolvidos com o OR-Tools:
  - Roteamento de Veículos: Encontrar as melhores rotas para uma frota de veículos que coletam e entregam pacotes dadas algumas restrições ("um caminhão não pode carregar mais do que 23 toneladas", "todas as entregas devem ser feitas dentro de uma janela de 8 horas", etc)
  - Escalonamento: Encontrar o escalonamento (agendamento) ótimo para um conjunto complexo de tarefas, sujeito a algumas restrições ("algumas atividades precisam ser executadas antes de outras atividades", "algumas atividades podem compartilhar um mesmo recurso", etc)
  - *Bin packing*: Empacotar vários objetos pequenos em recipientes, minimizando o número de recipientes necessários.

OR-Tools

OR-Tools inclui resolvedores:

- Programação por Restrições.
- Programação Linear.
- Programação Linear Inteira Mista.
- Roteamento de Veículos.
- Algoritmos em Grafos (caminho mais curto, fluxo de custo mínimo, fluxo máximo, etc)

OR-Tools

OR-Tools é escrito em **C++**, mas também pode ser usado com Python, Java ou C#.

- Link para instalação: `https://developers.google.com/optimization/install/cpp`
- Meu sistema é o Linux Mint 19.3 Tricia, então baixei a versão do OR-Tools para Ubuntu 18.04
- Link para vários exemplos: `https://developers.google.com/optimization/examples`

OR-Tools

$$\begin{aligned} \text{maximize} \quad & 3x + y, \\ \text{sujeito a} \quad & x + y \leq 2, \\ & 0 \leq x \leq 1, \\ & 0 \leq y \leq 2. \end{aligned}$$

$$\begin{aligned} \text{maximize} \quad & 3x + y, \\ \text{sujeito a} \quad & x + y \leq 2, \\ & 0 \leq x \leq 1, \\ & 0 \leq y \leq 2. \end{aligned}$$

```
#include "ortools/linear_solver/linear_solver.h"
using namespace operations_research;
int main() {
  // Criar o Resolvedor, que usa o GLOP
  MPSolver solver("simple_lp_program",
                  MPSolver::GLOP_LINEAR_PROGRAMMING);

  // Criar as variaveis x e y, ja dizendo o tipo,
  // os limitantes, e um nome
  MPVariable* const x = solver.MakeNumVar(0.0, 1, "x");
  MPVariable* const y = solver.MakeNumVar(0.0, 2, "y");
```

$$\begin{aligned} \text{maximize} \quad & 3x + y, \\ \text{sujeito a} \quad & x + y \leq 2, \\ & 0 \leq x \leq 1, \\ & 0 \leq y \leq 2. \end{aligned}$$

```
// Criar a restricao, 0 <= x + y <= 2.
MPConstraint* const ct = solver.MakeRowConstraint(0.0,
2.0, "ct");
ct->SetCoefficient(x, 1);
ct->SetCoefficient(y, 1);
// Criar a funcao objetivo, 3 * x + y.
MPObjective* const objective = solver.MutableObjective();
objective->SetCoefficient(x, 3);
objective->SetCoefficient(y, 1);
objective->SetMaximization();
```


$$\begin{aligned} \text{maximize} \quad & 3x + y, \\ \text{sujeito a} \quad & x + y \leq 2, \\ & 0 \leq x \leq 1, \\ & 0 \leq y \leq 2. \end{aligned}$$

```
  // Resolver!
  solver.Solve();

  std::cout << "Solution:" << std::endl;
  std::cout << "Objective value = " << objective->Value()
                                    << std::endl;
  std::cout << "x = " << x->solution_value() << std::endl;
  std::cout << "y = " << y->solution_value() << std::endl;
  return EXIT_SUCCESS;
}
```



```
hokama@fanatico:~$ cd /opt/or-tools_Ubuntu-18.04-64bit_v7.6.7691/
hokama@fanatico:/opt/or-tools_Ubuntu-18.04-64bit_v7.6.7691$ make run SOURCE=/home/hokama/cic111/teste.cc
g++ -fPIC -std=c++11 -O4 -DNDEBUG -Iinclude -I. -DARCH_K8 -Wno-deprecated -DUSE_CBC -DUSE_CLP -DUSE_BOP -DUSE_GL
OP \
 objs/teste..o \
 -Llib -Llib64 -lprotobuf -lglog -lgflags -lCbcSolver -lCbc -lOsiCbc -lCgl -lClpSolver -lClp -lOsiClp -lOsi -lCo
inUtils -lortools -Wl,-rpath,"\$ORIGIN" -Wl,-rpath,"\$ORIGIN/../lib64" -Wl,-rpath,"\$ORIGIN/../lib" -lz -lrt -lp
thread \
 -o bin/teste
bin/teste
Solution:
Objective value = 4
x = 1
y = 1
hokama@fanatico:/opt/or-tools_Ubuntu-18.04-64bit_v7.6.7691$
```